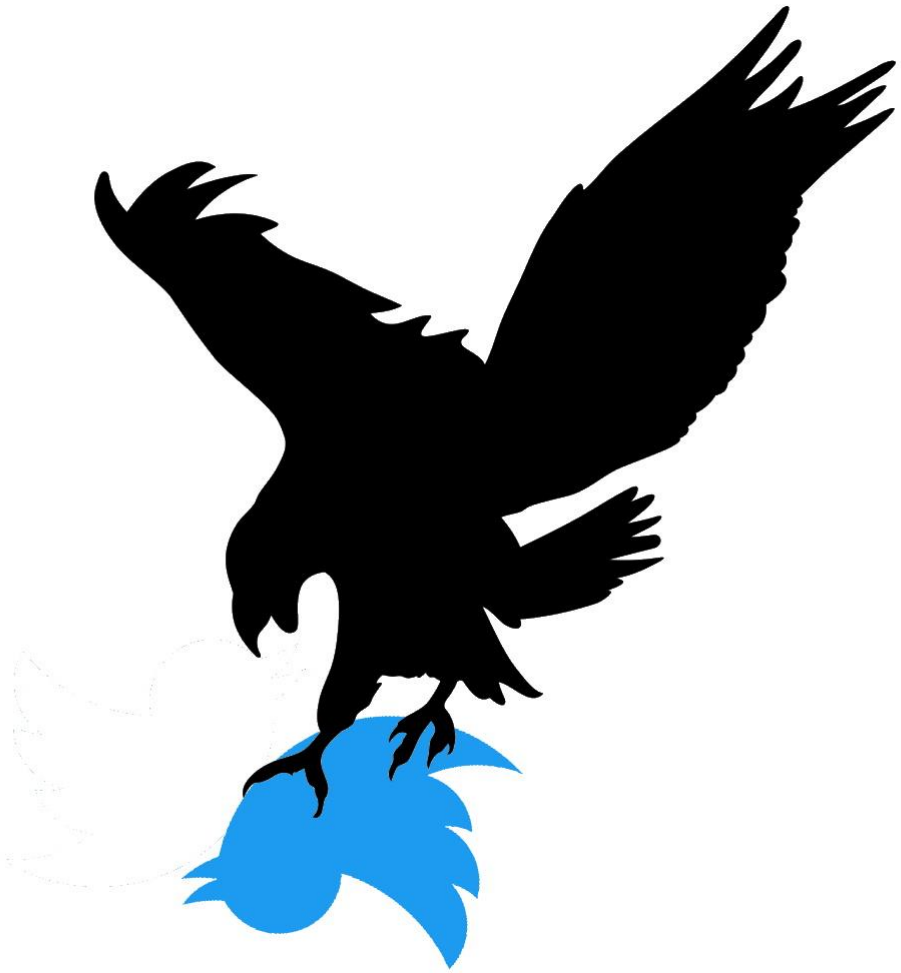
Croaker Forum



Projektdokumentation Modularbeit 151

Enea Siess IAP20-24A

24.10.2022

Ablauf des Implementierungs Prozess:

Da eines meiner wenigen Stärken das Prokrastinieren ist, setzte ich mich als erstes an das Croaker Logo. Nach einem 10-minütigen Design Prozess mit paint.net, war die Arbeit für den Tag getan, und ich konnte mich mit gutem Gewissen für die letzten 20 Minuten des Unterrichts ans Handy setzen und mich von Memes berieseln lassen. Eine Woche darauf, begann ich dann mit der Arbeit. Erst kopierte ich alle bisher bearbeiteten Files und lud sie in ein Git Repository. Darauf begann ich, die Files zu bearbeiten und implementierte options.php. Auf dieser Seite kann der Benutzer sein Passwort oder seinen Namen ändern. Auch eine Löschung des Accounts ist möglich. Danach war auch wieder Schluss für den Tag und ich nahm mir für die Ferien einiges vor.

In der zweiten Woche war es dann so weit, ich fuhr nach Österreich und machte – nichts. Am letzten Tag vor der Heimreise aber, implementierte ich die changepasswort.php- und changeusername.php Seite, allerdings war die Logik noch nicht vervollständigt. Nun hatte ich eineinhalb Wochen Zeit, das Projekt fertig zu stellen, allerdings machte mir mein Geschäft mit einer neuen Antiviren Software einen Strich durch die Rechnung. Ich konnte die 4 Lektionen im Unterricht nicht nutzen und verprasste noch einmal 3 Stunden zuhause, während ich hilflos versuchte eine VM, die nicht instantan abstürzt aufzusetzen. Frustriert gab ich dann auf, und überliess meinen Laptop zum Aufsetzen der VM am Donnerstag der firmeninternen IT – welche es auch nicht schaffte

Nun war Donnerstagabend und ich hatte nichtmehr allzu viel Zeit, mein Projekt fertig zu stellen. Ich entschied mich also dazu, meinen PC wieder zusammen zu schrauben und zu installieren, was auch mehr oder weniger meinen gesamten Abend frass. Freitag ist Freitag; impliziert KEINE SCHULE! Also musste ich den Samstag damit verbringen, die Datenbank richtig anzupassen, die Logik für das Ändern des Benutzernamens und des Passworts zu implementieren und einen Post Button zu designen. Am Ende des Tages, war es dem Benutzer möglich einen Croak zu erstellen, welcher aber noch nirgendwo sichtbar war.

Nun war Sonntag, nur noch 2 Tage bis zur Abgabe und am Montag musste ich lange arbeiten, bedeutete, ich musste den gesamten Tag für mein Project einplanen. Um 13 Uhr stand ich auf, startete meinen Computer, und setzte mich an die Arbeit. Ich kam voran, implementierte den Croakerfeed und die Seite, auf der die eigenen Croaks einsehbar sind. Allerdings musste ich mich meinem Todfeind, HTML-CSS wieder stellen! Dementsprechend gross, war die Frustration, als ich 2 Stunden verschwendete, um den Editbutton auf den eigenen Croaks anzeigen zu lassen. Ich implementierte die Seite croak.php und die gesamte Logik und abgesehen von wenigen Validierungsfehlern war das Project um 11 Uhr Abends eigentlich Abgabe bereit. Im Flow wie ich war, habe ich natürlich keine Backus auf GitHub hochgeladen und comittete alle Changes am Montag Abend, als ich gegen 7 Uhr von der Arbeit nach Hause kam.

Ich musste nun nur noch die Server- und Clientseitige Validierung implementieren und den HTML-CSS Code validieren lassen. Allerdings habe ich nicht damit gerechnet, wie sehr ich an REGEX verzweifeln würde. Inklusive Abendessen war ich um 10 Uhr mit der Server und clientseitigen Validierung fertig und hatte nun noch die Projektdokumentation, das

fertigstellen des Lerntagebuches, das Erstellen des readme.md Files und den mysql-dump vor mir. Ich begann mit der Codevalidierung, konnte jedoch einige Errors nicht richtig deuten, weshalb ich das validieren des Codes, trotz der hohen Punkte Gewichtung, vorerst auf die Seite schob.

Projektantrag:

Ausgangssituation / Problembeschreibung:

Sind Sie genervter Twitteruser? Gelangweilt von der Einseitigkeit der Algorithmus-Bubble? Frustriert von dem stumpfsinnigen Populismus, der auf Twitter verbreitet wird, der 150-Zeichen Politik und der zunehmenden Extremisierung auf dieser Socialmedia Plattform? Denken Sie, Sie sollten Twitter endgültig löschen, sich wieder den wichtigen Dingen im Leben widmen und die Socialmediaidioten hinter sich lassen? Falsch! Sieß Ltd. präsentiert ihnen hiermit “Croaker”, das Forum für Idioten mit höherem Anspruch! Lassen Sie Ihren Frust raus, beleidigen Sie politisch Andersdenkende auf einer höheren intellektuellen Ebene und pushen Sie Ihr Selbstwertgefühl erneut hoch, wie in den guten alten Zeiten! Auf “Croaker” ist es ihnen möglich, eine eigene Seite über ihre politischen Standpunkte zu erstellen, anonym in Kommentaren zu beleidigen und Beiträge mit anderen zu verfassen! Sie wissen, wer im Besitz von ihren Daten ist und wer sie weiterverkauft (Enea Siess), und müssen nicht mehr auf die "Amis" oder die “Russen” spekulieren! Da die Software so primitiv ist, kreiert auch kein Algorithmus, der ihre Meinungsbildung beeinflussen könnte, ihre For-you Page.

Croaker must haves:

- Die Möglichkeit einen Account zu erstellen
- Die Möglichkeit sich an- und abzumelden besteht
- Der HTML und CSS code ist validiert
- Benutzereingaben werden beidseits validiert und Injections jeglicher Art unterbunden
- Durch Session-Handling hat ein eingeloggter User die Möglichkeit, einen Beitrag zu verfassen und Kommentare zu Schreiben
- Ein nicht eingeloggter User kann anonyme Kommentare verfassen
- Passwörter werden gehasht und gestaltet
- Benutzernamen und Passwort lassen sich im nachhinein verändern
- Dem User ist es möglich einen eigenen Beitrag zu bearbeiten oder zu löschen
- Session-Fixation und Session-Hijacking wird erschwert

Croaker nice to haves:

- Dem User ist es möglich, Kommentare verfassen
- Dem User ist es möglich, anonyme Kommentare verfassen
- Dem User ist es möglich, ein Profilbild festzusetzen
- Dem User ist es möglich, einen Beitrag mit einem anderen User zu verfassen

Projektgesamtziel:

C1 : Wurde mit diesem Antrag erfüllt
C2 : Ich werde Trello für die Planung dieses Projekts verwenden
C3 : Ich werde sobald ich mit dem Projekt und dessen Planung beginne, ein Google Docs Dokument
erstellen und in diesem den Developmentprozess meines Projektes dokumentieren.

C4 : in einer Readme.txt Datei im Git Verzeichnis wird eine Installationsanleitung hinterlegt
C5 : Ich werde Github verwenden um meinen Code zu versionieren und das Projekt zu backuppen
C6 : HTML Validator: validator.w3.org CSS Validator: jigsaw.w3.org/css-validator/

Projektgruppe:

Enea Siess

Projektauftraggeber/in:

Daniel Brodbeck

Projektdauer:

Geplanter Beginn: 13.9.2022
Geplantes Ende: 18.10.2022
Zur Verfügung stehende Lektionen in der Schule: 16

Projektplanung:

https://trello.com/invite/croaker3/ef71474635606f4c42f308dd39d73d71

Projektressourcen:

Ressourcen	**Beschreibung**
Personal:	Enea
Entwicklungsumgebung:	VS-Code
Programmiersprachen:	PHP, HTML, CSS
Geräte:	Laptop Dell Latitude 7300

Projektrisiken und-unsicherheiten:

Projektrisiken:

- Nicht rechtzeitige Fertigstellung durch unterschätzte Komplexität des Projektes
- Verzögerung durch mangelnde PHP und MYSQL-Kenntnisse
- Unschöne Gestaltung durch schlecht umgesetzte Textdarstellung und Texthinterlegung in der Datenbank

Projektantrag Reflektion:

Abgesehen von der Validierung des HTML-CSS Codes, wurden bis zum jetzigen Zeitpunkt alle must haves erfüllt. Da mir die Programmierung des Projekts, trotz vielen Rückschlägen, viel Spass gemacht hat, hätte ich sehr viel Freude daran gehabt, die nice to haves auch noch zu implementieren. Allerdings ist in 1:36h Abgabe und ich habe noch einiges vor mir. Von den Projektrisiken ist überraschenderweise nur eines aufgetreten: Das Design. Umbrüche werden in den einzelnen Croaks nicht angezeigt. So wie ich mich kenne, werde ich das Projekt leider ohne den Notendruck nie fertigstellen...

Kompetenz C3, Projektdokumentation:

Wird mit diesem Dokument vervollständigt.

Kompetenz C4, README.md:

Die genaue Installationsanweisung ist auf GitHub als README.md gespeichert.

Kompetenz C5, Versionierung:

Mein Projekt wurde lückenfrei, aber nicht allzu regelmässig auf GitHub versioniert. Das Repository ist öffentlich unter https://github.com/Fridadacat/CroakerForum einsehbar.

Kompetenz C6, Validierung HTML & CSS:

Nach der Vervollständigung meiner Dokumentation kehre ich nun wieder zu diesem Abschnitt zurück. Ich habe nur noch 38 Minuten Zeit bis zur Abgabe, deshalb muss ich leider aus Zeitgründen die Validierung grösstenteils weglassen. Einige Fehler, die durch den w3c Validator ausgestrichen wurden, wurden in index.php trotzdem noch behoben. Ich hoffe ich kann ihnen somit trotzdem meine Kompetenz, Htmlcode in den Validator zu copypasten und die Fehler zu beheben, vermitteln und zumindest noch ein paar Teilpunkte erhaschen. Schlechtes Time Management war Schuld an der Sache, obwohl ich da auch meiner Firma eine gewisse Schuld zuschiebe.

Kompetenz C7, Benutzerseitige Validierung:

Alle Eingaben werden benutzerseitig validiert. Gemacht wurde dies meist mit dem HTML-Form Pattern Attribut. Folgend ein Codebeispiel der Passwort Validierung mit regex:

```
placeholder="Gross- und Kleinbuchstaben, Zahlen, Sonderzeichen, min. 8 Zeichen, keine Umlaute"
pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$"
```

Auch die Croaks werden auf fehlenden Input geprüft.

Kompetenz C8, Serverseitige Validierung:

Alle Eingaben werden Serverseitig validiert. Falls jemand manuell an der clientseitigen Validierung herumschraubt, prüft der Server noch einmal auf Fehler. Hier das Serverseitige Regex Beispiel zur Passwortvalidierung:

```
$password = trim($_POST['password']);
// passwort gültig?
if(empty($password) || !preg_match("/(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/", $password)){
    $error .= "Das Passwort entspricht nicht dem geforderten Format.<br />";
}
```

Kompetenz C9, Script injection wird in meinem Projekt verhindert:

Alle Benutzereingaben, die später an ein SQL-Statement gebunden werden, werden als Parameter mit bind_param() übergeben. Folgend ein Codebeispiel zu einem SQL der mit dem Benutzernamen das gehashte Passwort requestet:

```php
$query = "select password from user where username = ?";
$stmt = $mysqli->prepare($query);
$stmt->bind_param("s", $username);
$stmt->execute();
$result=$stmt->get_result();
$dbPassword = $result->fetch_assoc();
```

Auch Scriptinjection auf Seite des Browsers wird verhindert. Z.B. wird die Ausführung eines Scripts durch einen Croak verhindert:

PapstFranziskus

<script>alert('Hello World')</script>

2022-10-23 21:18:09

Dies wird durch die Ausgabe des Croaks in uml bewerkstelligt. Somit interpretiert der Browser den uml Text wieder zu normalem, und führt den Code nicht aus. Dies wird mit der sanitize Methode gemacht, welche ich bei der Anzeige eines Croaks und beim Bearbeiten eines Croaks implementiert habe:

```php
$croak = filter_var($croak, FILTER_SANITIZE_SPECIAL_CHARS);
```

Kompetenz C10, Sessionhandling:

In meinem Projekt kann ein angemeldeter User auf sein Userhome zugreiffen, cCoaks erstellen und die Optionen aufrufen. Dies bleibt den nicht angemeldeten Usern verwehrt.

Kompetenz C11, Logout:

Der Benutzer kann in der Menüleiste unter “Logout” abmelden, wobei die Session sofort geschlossen wird. Folgend die Codestelle:

logout.php

```php
<?php
    session_start();
    $_SESSION = array();
    session_destroy();
    header('Location:login.php');
?>
```

Kompetenz C12, Session fixation & Hijacking:

Sessionfixation wird erschwert, da nach jedem Login eine neue Sessionid generiert wird. Dies wird mit folgender Methode gemacht:

```
session_regenerate_id(true);
```

Sessionhijacking wird insofern unterbunden, dass der User sich alle 24 Minuten, oder nach schliessen des Browsers, wieder einloggen muss. 24 Minuten ist die standardisierte Zeit im php.ini File, welche unter folgendem Parameter in Sekunden angegeben wird: session.gc_maxlifetime = (Sekunden).

Kompetenz C13, hash und salting:

Bei der Festlegung des Passworts (Registrierung oder manuelle Änderung), wird das Passwort gehasht und gesaltet. Dies geschieht mit der password_hash() Methode. Folgend ein Code Beispiel:

```
password_hash($password, PASSWORD_DEFAULT);
```

Das Passwort wird auch nur encrypted ausgelesen und kann mit dem Klartext Passwort mit der Methode password_verify() verglichen werden. Dies habe ich beim Login, beim ändern des Benutzer namens und beim ändern des Passworts implementiert:

```
password_verify($password, $dbPassword["password"])
```

Kompetenz C14, Registrierung:

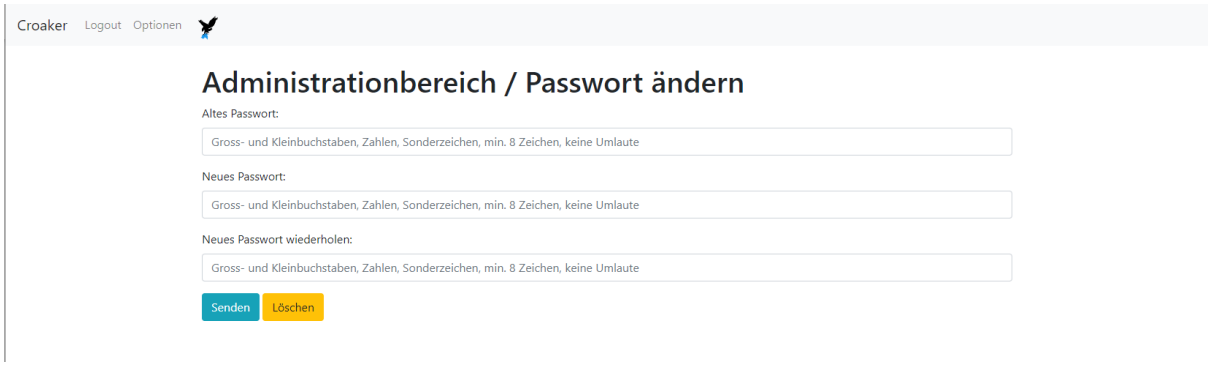
Der Benutzer kann sich unter register.php auf Croaker registrieren. Dabei wird der Namen, der Nachnamen, die Mail, der Benutzername und das Passwort des Benutzers gespeichert. Leider wird in der jetzigen Version des Projekts nur der Benutzername und das Passwort verwendet.

Kompetenz C15, Registrierung:

Der Benutzer kann sich unter login.php anmelden. Dabei werden sein Benutzername und sein Passwort abgefragt. Ist das Passwort korrekt, erhält er Zugriff auf den Myspace, die Optionenseite, die Logoutseite und kann Croaks verfassen, bearbeiten und löschen.

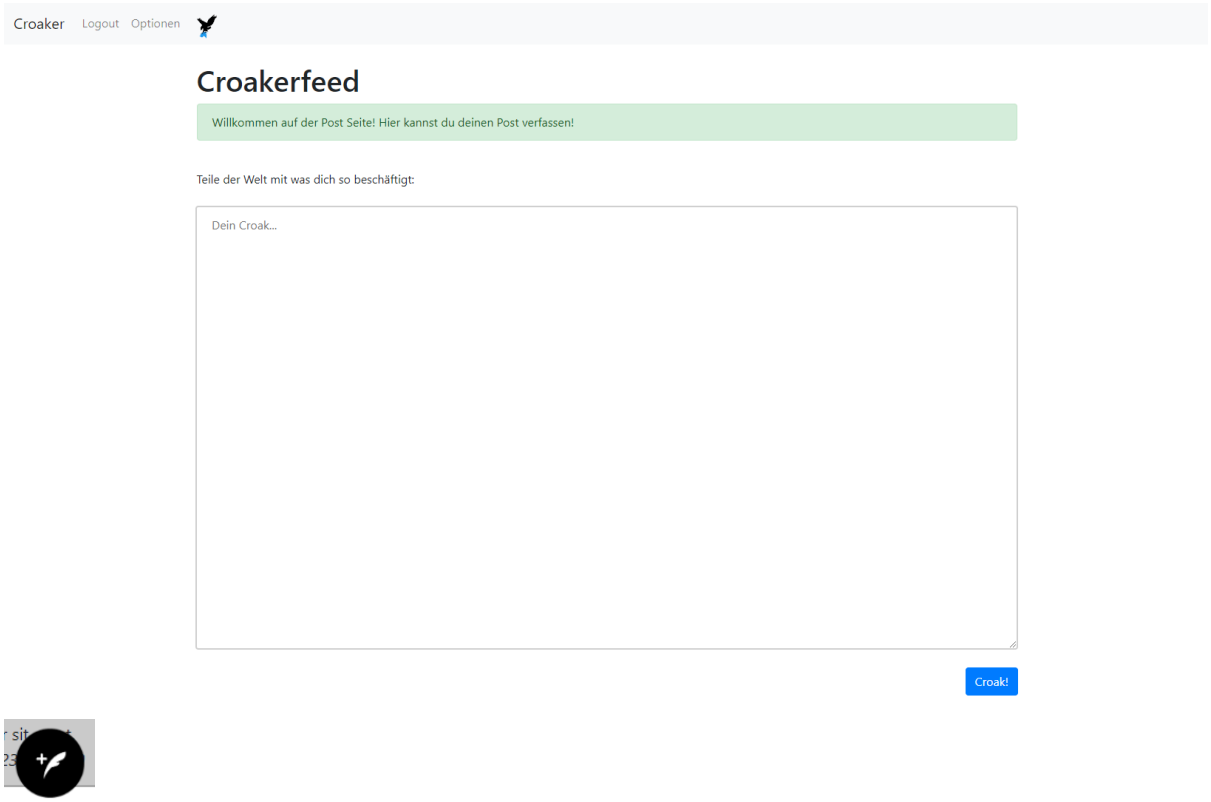
Kompetenz C16, Änderung des Passworts:

Unter Optionen => Passwort ändern, kann ein Benutzer sein Passwort ändern. Dabei wird sein altes Passwort und sein neues Passwort doppelt abgefragt. Ist die Validierung Beider korrekt, wird das Passwort geändert. Die Benutzerschnittstelle sieht so aus:



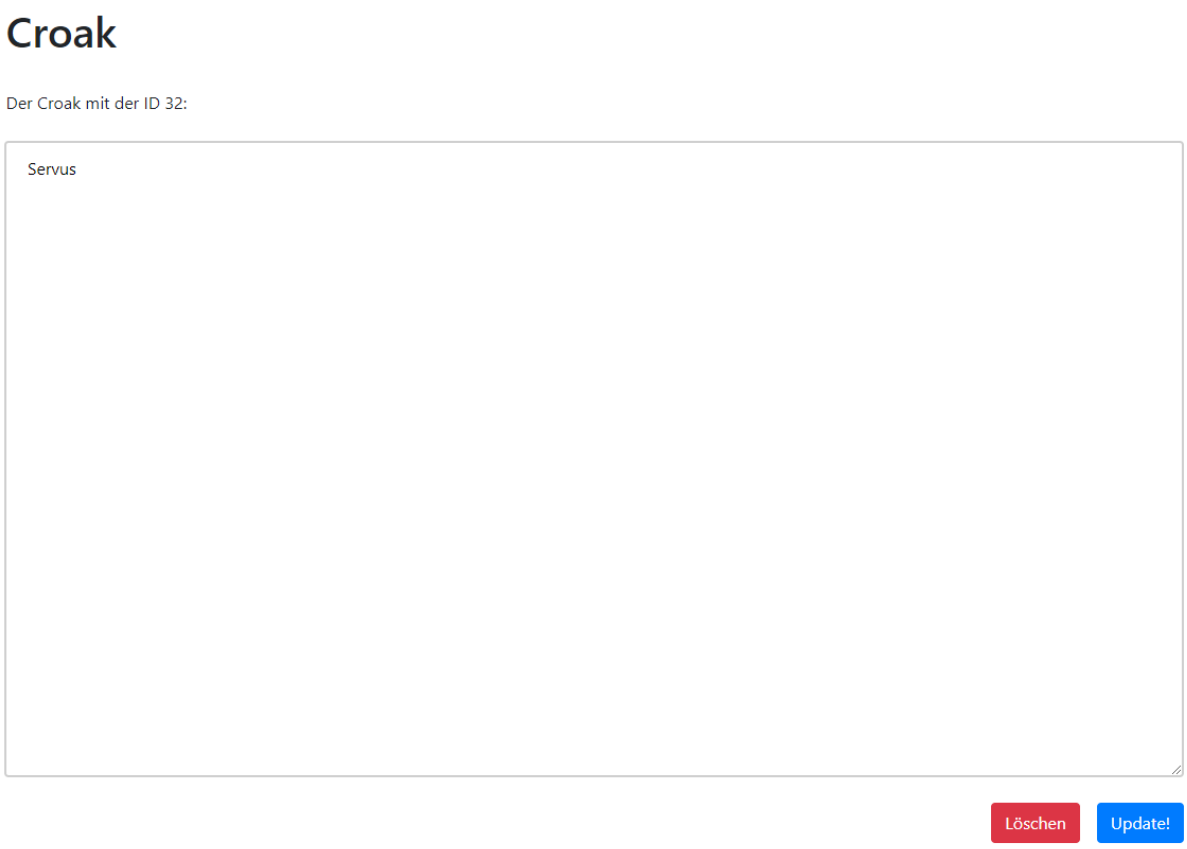
Kompetenz C17, Croaken:

Ein angemeldeter User kann auf dem Croakerfeed sowie auf seiner eigenen Seite den “croak-button” im unteren Bildrand rechts verwenden und einen Croak verfassen. Dieser wird gespeichert, und anschliessend im Croakerfeed angezeigt. Folgend ein Bild vom Croakbutton und der post.php Seite:



Kompetenz C18, Croaken:

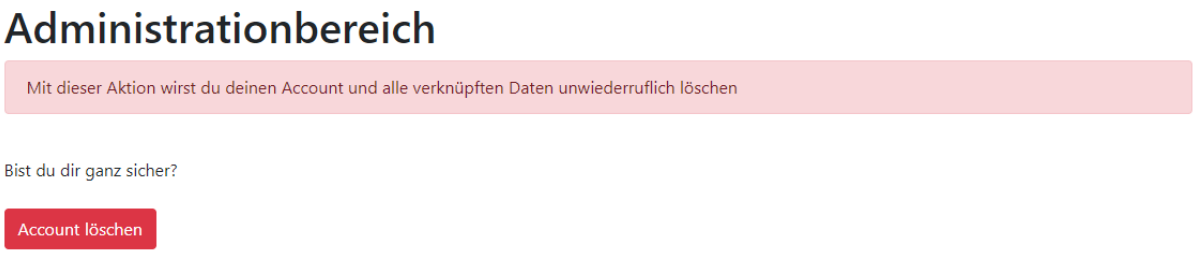
Wenn ein Benutzer auf einen eigenen Croak klickt,, kann er seinen Croak bearbeiten sowie löschen. Dabei wird seine Identität geprüft und dementsprechend die Buttons sowie das Textfeld zur Verfügung gestellt:



Gehört der Croak nicht dem Benutzer, wird ihm nur ein Melden Button angezeigt:

Kompetenz C19, Un-Croaken:

Wie bereits in C18 erwähnt, kann ein Benutzer seine eigenen Croaks löschen. Ist er aber so frustriert ab der Inaktivität der Croakercommunity, dessen einzigen Mitglied er selbst ist, kann er auch unter Optionen => Account löschen, alle Seine Croaks inclusive seinen Account löschen.



Kompetenz C20, SQL-Injection:

Wie bereits in C9 erwähnt, wird jeder User Input erst nach dem Ausführen des SQL-Statements als Parameter übergeben. Somit ist es dem Benutzer nicht möglich, Veränderungen an der Datenbank vorzunehmen: